AFRL-IF-RS-TR-2005-302
Final Technical Report
August 2005

DISTRIBUTED COLLABORATION ENVIRONMENT FOR RESEARCH (DYCER)

Shinkuro, Incorporated

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. N883

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

AFRL-IF-RS-TR-2005-302 has been reviewed and is approved for publication

APPROVED: /s/
JON B. VALENTE
Project Engineer

FOR THE DIRECTOR: /s/
WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** AUGUST 2005	**3. REPORT TYPE AND DATES COVERED** Final Aug 02 – Mar 05

4. TITLE AND SUBTITLE DISTRIBUTED COLLABORATION ENVIRONMENT FOR RESEARCH (DYCER)	**5. FUNDING NUMBERS** C - F30602-02-C-0204 PE - 62301E PR - N883 TA - 01 WU - 04
6. AUTHOR(S) Stephen D. Crocker	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Shinkuro, Incorporated 5110 Edgemoor Lane Bethesda Maryland 20914	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington Virginia 22203-1714; AFRL/IFGA 525 Brooks Road Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2005-302

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Jon B. Valente/IFGA/(315) 330-1559/ Jon.Valente@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

The DyCER project was an experiment in using file sharing as an integral part of network-based collaboration environment. During the course of the project, we tested the concepts in many different scenarios and used the feedback to refine the system. As a result, the software became more efficient and suited to a broader class of real-world applications.
The ability to share information securely and quickly in groups with dynamic membership is a powerful capability that can become the building block for new classes of applications. The work done under DARPA funding has demonstrated that military, government, and business users can benefit immediately from DyCER's enhanced architecture and implementation.

14. SUBJECT TERMS Groups, Security, Group File Sharing, File Replication, Cross-Platform, Group Screen Sharing	**15. NUMBER OF PAGES** 31
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

1.0 Summary

DyCER is the Dynamic Collaboration Environment Research system under development by Shinkuro, Inc. It is designed to support rapid formation of coalitions and persistent collaboration within the coalitions under a wide variety of security and communication environments. The initial software development occurred with private support. Ongoing refinement for this range of security and first responder situations has been supported by the Defense Advanced Research Projects Agency, Advanced Technology Office, under DARPA N883, Contract F30602-02-C-0204. This paper is the final report on the project, describing the thinking behind DyCER, its architectural components, and the results of a public beta release.

The DyCER project was an experiment in using file sharing as an integral part of network-based collaboration environment. During the course of the project, we tested the concepts in many different scenarios and used the feedback to refine the system. As a result, the software became more efficient and suited to a broader class of real-world applications.

The ability to share information securely and quickly in groups with dynamic membership is a powerful capability that can become the building block for new classes of applications. The work done under DARPA funding has demonstrated that military, government, and business users can benefit immediately from DyCER's enhanced architecture and implementation.

2.0 Introduction

People are collaborative by nature, but the Internet is not. People talk and listen in groups as part of their everyday life, but the Internet is based on data connections between two machines. If the Internet is to support collaboration, the applications that run on it must be based on group communication principles. DyCER is both an application and a framework for secure group communication.

DyCER was inspired by a DARPA-funded project for secure file sharing. TRUFFLES was a project that began by modifying a Unix-based operating system kernel to support network file sharing based on group permissions. Although Berkeley Unix had long offered the Network File System protocol (NFS), it was oriented towards local area networks and file systems defined by system administrators. TRUFFLES was to have smaller units of sharing (files instead of file systems) and user-defined group membership. During the period of its development, TRUFFLES began running up against barriers on the Internet.

The evolution of the Internet from its original form to today's environment has shifted the way we think about deploying shared-information solutions. The early days of the

Internet (until about 1990) were marked by open access to virtually all systems using the Internet address space. The rapidly increasing system administration complexities, together with security concerns, led to the widespread deployment of firewalls, while concern about address availability resulted in requirements to use ephemeral addresses assigned by gateways running Network Address Translation (NAT). Within a few years these changes forced a new view of what an Internet application should be.

These events are important because they fundamentally changed the nature of connectivity on the Internet, and this made some kinds of collaboration tools impossible. No longer can any two machines open a socket between each other for sharing data or for conversation. NAT and firewall technology make it difficult for machines to advertise their availability and to accept data connections. As a result, ordinary users cannot use their own machines to offer Internet services, even to a small, selected audience.

The success in the mid 1990's of the World Wide Web and HTTP (Hypertext Transport Protocol) established the standard way to share information. Servers must have a presence outside a firewall and must have fixed IP (Internet Protocol) addresses, while clients must be behind firewalls and have ephemeral addresses.

Collaboration tools do exist, despite these retarding obstacles. Almost every computer connected to the Internet can open a connection to a hosted web server, and therefore many of the collaboration systems are implemented on web servers. These collaboration services can bring address books, bulletin board, file sharing, discussions, project management, and other typical collaboration tools together into a hosted solution.

While hosted solutions support communication among people at different companies and do not require IT installation and support, they suffer from limitations of scaling and security. With respect to scaling, most web-based collaboration systems require that all users connect to the same server. Any ordinary server has natural limitations on its capacity. In order to serve very large numbers of users, the design of the server becomes more complex as it is scaled up. Some method is needed to designate the focal point for of the users involved in a specific collaboration. A simple approach is to have the users choose a specific collaboration server, but this strategy places a burden on the users, is difficult to coordinate, and doesn't guarantee the loads on the servers will be balanced. Alternatively, the hosting service can redirect the connections to various servers, but this adds considerable complexity and cost to the server structure. There is no solution for the user who wants a small-scale, inexpensive, and easily maintained collaboration service.

In addition to issues of functionality and scaling, the location of intellectual property is another concern for users of hosted solutions. In general, companies are not comfortable with moving sensitive information outside of their control. Hosting companies have to provide strong assurances to their clients that information hosted on their machines is protected and will not be misused. These assurances increase the cost and complexity of the hosted solutions. To counter this problem, some of the hosted solutions, like eRoom, offer sales of their servers to enterprises. While that solution sometimes provides a good

Intranet solution, it places the organization that purchases the server in the same business as the hosted provider and requires that the purchasers make their collaborative servers accessible on the Internet for any work between organizations. It also creates a single point of failure, because if the collaboration server fails, all of the data is inaccessible until the server is restored from backup.

A second approach based on peer-to-peer (P2P) technology emerged in late 1990s to early 2000. Groove Networks, Endeavors Technology, Roku, and others created a means for sharing information without requiring that all information be saved on a central, hosted server. These companies focus on direct connections between individual client systems and offer either access to files or replication of files. However, these companies also needed to cope with firewalls and address translation (NAT) and they created a switch of sorts: a system that clients could connect to using an outbound connection that then routes requests between other connected systems.

Although these solutions do solve the problem of crossing the firewalls, two new problems emerged. First, scale is again an issue because these "switches" must support connectivity between all of their users. As leaders in the Instant Messaging field have found, there is a significant amount of engineering work that must go into minimizing latency in multi-user communication. This work is no simpler in the peer-to-peer case than in the dedicated server case.

A second problem also exists. Client computers systems do not have the same operational characteristics that servers do. They are often turned off on a regular basis. They may not ever have the same IP address and may shift from network to network. They will also have varying bandwidth. Mobile users may have high speed Internet at the office but dial-up from the road (or in the case of some of us, higher bandwidth through the hotel's Internet line than we have at home). Thus, the performance of direct connections between systems varies greatly, making it very difficult to engineer systems with reliable and minimal response times.

Shinkuro looked at these problems and decided that different approach was needed. The solution needed to have the following characteristics:

- Massive scale
- Connectivity between all of the users, even those behind firewalls or NAT
- Transparent file sharing
- Handling of disconnected users
- No new servers on the Internet

One seemingly obvious requirement that is not on the list is instant communication between users. This is a requirement that most of the other systems believe is necessary. None of the typical collaborative tools (file synchronization and replication, message boards, discussion groups, group calendaring, "to do" lists, etc.) requires instant communication. The focus of our innovation is on data sharing; instant communication is important, but it is an adjunct technology that enhances the experiences of users of our core technology.

As we considered the design for a protocol for information sharing in a dynamic group, the advantages of a store-and-forward protocol stood out. The Internet email protocol, SMTP, is such a protocol, and it has proven to be a good way to move messages reliably among millions of users. It is easy to get an email address, and it is easy to send email to anyone who has one.

Shinkuro implemented and distributed application software that used SMTP in DyCER Version 1 (V1). After distribution of V1, we considered the user comments and experiences and redesigned the communications protocol. That was incorporated into DyCER Version 2 (V2). As users told us how they were using DyCER to share information, we understood that they needed new capabilities and new ways to organize their group relationships, and we incorporated these changes into V2. In the following sections, we describe V1 and V2 and discuss the implications of our findings.

3.0 Methods

The methods for evaluating and refining the software have been based on encouraging use by distributing the binary software freely and soliciting feedback. Also, our onsite participation in the JWID exercise gave us direct experience with scripted demonstrations tailored to cybersecurity response teams.

The Shinkuro website at http://www.shinkuro.com has a simple interface for downloading the software, and there is also a support forum for feedback and help. Version 1 of the software was available in during the first quarter 2003, and the feedback resulted in the development of Version 2, which was first available in March of 2004.

In order to encourage use and to get more feedback, the software runs on multiple OS/hardware platforms, and this has proven to be important for collaborative group formation. By removing the platform obstacle, we make it much easier for users to collaborate without prior planning.

We have also sought out "power users" who are familiar with collaboration tools and have the social contacts to promote group formation. Through interviews and written feedback we have been able to refine the software for usability and scalability.

Version 2 brought the secure relay capability, and we established a free relay for general users at shinkuro.com. The relay has been very helpful in exploring the communications scalability of the software and in providing statistics for "tuning" the software.

The JWID exercise in 2004 and the follow-on CWID exercise in 2005 were particularly important because they exercised all aspects of the software and involved more complex communication patterns than casual usage. The "lessons learned" resulted in ongoing changes to the software, which has been continuously evolving and improving.

4.0 Results and Discussion

4.1 DyCER Version 1 (V1)

4.1.1 Overview

DyCER is a group formation and information sharing system designed to simplify the task of sharing data among users of computer systems. It has several objectives:

- Security
- Overcoming the barriers inhibiting direct connectivity between computer systems
- Supporting participants in different administrative domains
- Sharing communications but not infrastructure
- Flexible group formation strategies
- Cross-platform operation
- Creating a development platform

The DyCER system focused initially on a group file sharing capability based on a peer group formation protocol. Group file sharing was implemented using file replication – group members would contribute files to a group. These files would be replicated to the local disk of each of the other group members. The V1 system implemented this capability by implementing a message-passing system using e-mail (SMTP and mail retrieval protocols) as a transport mechanism.

4.1.2 Architecture

The DyCER Version 1 (V1) architecture was a message router with an application layer interface (Figure 1). V1 sat in the e-mail stream, extracting messages that were designated for the DyCER application. DyCER acted like a custom e-mail filter, except that messages were intercepted before they reached the e-mail client. As the messages arrived, the DyCER software quickly determined if DyCER should handle the message. If not, the messages were passed along to the normal e-mail client.

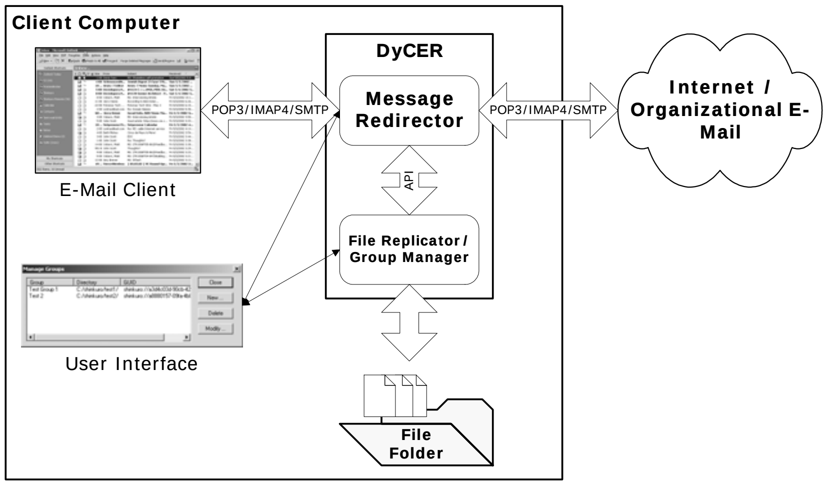


Figure 1 - Email redirection

Figure 1 shows the relationship between the message redirector and the e-mail stream. In particular, the message redirector sat in the e-mail stream and acted as a proxy for POP3 and SMTP. It also operated side-by-side with an e-mail client, polling the e-mail server periodically. It responded to calls from the e-mail client and connected directly. A separate interface allowed a user to set the parameters of the DyCER software.

The V1 system had two main layers – the application layer and the communications layer (Figure 2). The communications layer implemented the message redirector and collected the messages for the application layer. The application layer handled the messages in the context of whatever application it implemented. In the first version of DyCER, the application layer implemented the group file sharing application.

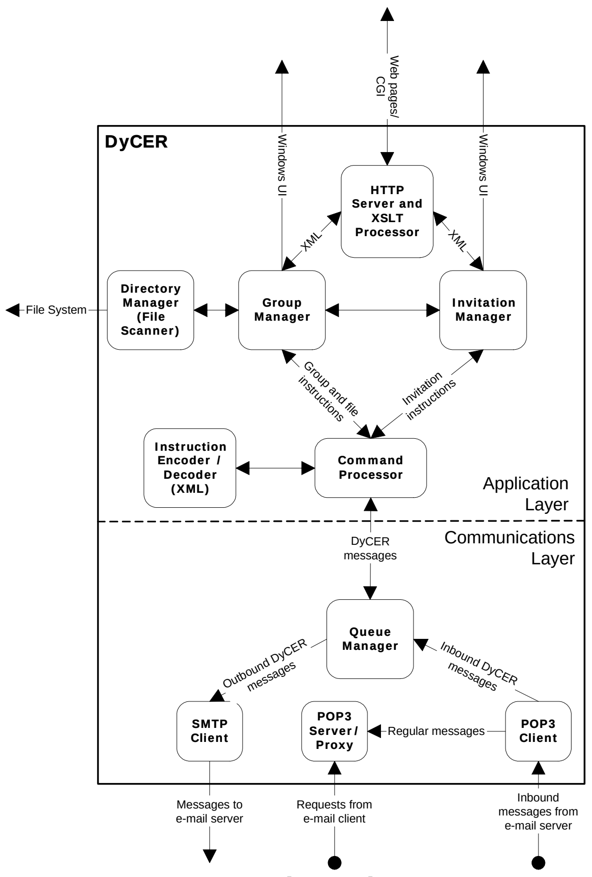
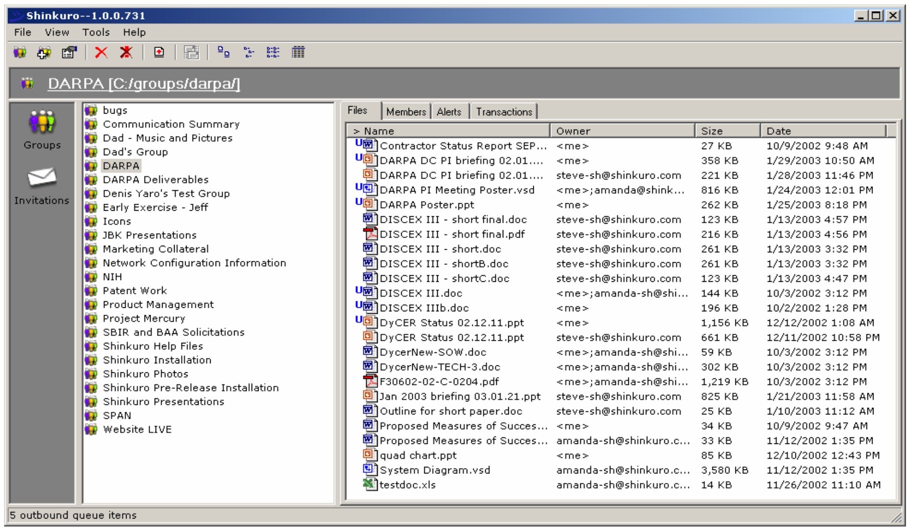


Figure 2 - Two-layer architecture

The V1 interface was designed to look similar to other Windows collaboration applications, in particular Microsoft Outlook (Figure 3). The left-most panel switched between groups and invitations. The center panel contained a list of groups while the right-most panel contained the detailed information for each group, including files, members, alerts and transactions.



- bugs
- Communication Summary
- Dad - Music and Pictures
- Dad's Group
- DARPA
- DARPA Deliverables
- Denis Yaro's Test Group
- Early Exercise - Jeff
- Icons
- JBK Presentations
- Marketing Collateral
- Network Configuration Information
- NIH
- Patent Work
- Product Management
- Project Mercury
- SBIR and BAA Solicitations
- Shinkuro Help Files
- Shinkuro Installation
- Shinkuro Photos
- Shinkuro Pre-Release Installation
- Shinkuro Presentations
- SPAN
- Website LIVE

> Name	Owner	Size	Date
U Contractor Status Report SEP...	<me>	27 KB	10/9/2002 9:48 AM
U DARPA DC PI briefing 02.01....	<me>	358 KB	1/29/2003 10:50 AM
DARPA DC PI briefing 02.01....	steve-sh@shinkuro.com	221 KB	1/28/2003 11:46 PM
U DARPA PI Meeting Poster.vsd	<me>;amanda@shink...	816 KB	1/24/2003 12:01 PM
U DARPA Poster.ppt	<me>	262 KB	1/25/2003 8:18 PM
DISCEX III - short final.doc	steve-sh@shinkuro.com	123 KB	1/13/2003 4:57 PM
DISCEX III - short final.pdf	steve-sh@shinkuro.com	216 KB	1/13/2003 4:56 PM
DISCEX III - short.doc	steve-sh@shinkuro.com	261 KB	1/13/2003 3:32 PM
DISCEX III - shortB.doc	steve-sh@shinkuro.com	261 KB	1/13/2003 3:32 PM
DISCEX III - shortC.doc	steve-sh@shinkuro.com	123 KB	1/13/2003 4:47 PM
U DISCEX III.doc	<me>;amanda-sh@shi...	144 KB	10/3/2002 3:12 PM
U DISCEX IIIb.doc	<me>	196 KB	10/2/2002 1:28 PM
U DyCER Status 02.12.11.ppt	<me>	1,156 KB	12/12/2002 1:08 AM
DyCER Status 02.12.11.ppt	steve-sh@shinkuro.com	661 KB	12/11/2002 10:58 PM
DycerNew-SOW.doc	<me>;amanda-sh@shi...	59 KB	10/3/2002 3:12 PM
DycerNew-TECH-3.doc	<me>;amanda-sh@shi...	302 KB	10/3/2002 3:12 PM
F30602-02-C-0204.pdf	<me>;amanda-sh@shi...	1,219 KB	10/3/2002 3:12 PM
Jan 2003 briefing 03.01.21.ppt	steve-sh@shinkuro.com	825 KB	1/21/2003 11:58 AM
Outline for short paper.doc	steve-sh@shinkuro.com	25 KB	1/10/2003 11:12 AM
Proposed Measures of Succes...	<me>	34 KB	10/9/2002 9:47 AM
Proposed Measures of Succes...	amanda-sh@shinkuro.c...	33 KB	11/12/2002 1:35 PM
quad chart.ppt	<me>	85 KB	12/10/2002 12:43 PM
System Diagram.vsd	amanda-sh@shinkuro.c...	3,580 KB	11/12/2002 1:35 PM
testdoc.xls	amanda-sh@shinkuro.c...	14 KB	11/26/2002 11:10 AM

5 outbound queue items

Figure 3 - Group file status

4.1.3 Important Lessons

The Shinkuro website had the V1 software available for easy download and installation on machines running the Windows OS starting in the spring of 2003. Advertising of the software by word-of-mouth to colleagues and some software reviewers, we got immediate feedback through our support forum and through email about problems, successes, and “wish lists”. There were several core findings from the V1 experience.

E-mail is an ideal namespace for identifying arbitrary Internet users. Virtually everyone on the Internet has an e-mail address, making it the largest organized namespace available. While users do occasionally change e-mail addresses as a result of shifts in employment or service provider, the large bulk of communications occurs on a relatively stable set of known names. Since individuals are accustomed to exchanging e-mail addresses, e-mail is a simple, viral way of spreading an application through the Internet.

E-mail protocols are problematic for file replication. For all of its benefits, e-mail has problems meeting the requirements of reliable, assured delivery of general content. E-mail servers often limit their users to a certain amount of disk storage for their inboxes. This limitation, when hit, results in an error or bounce back to the server attempting to transmit the message. Because an e-mail message may go through several hops before reaching its ultimate destination, a failed message may not ever be retried for delivery; instead it may be bounced back to the sender with a non-standard message (sometimes with the original message attached, sometimes not). This eliminates the ability of an automated system to do error recovery on failed messages. In addition, failed messages may not be intercepted automatically, resulting in additional clutter in the user’s inbox.

In addition to basic error handling and delivery difficulty, some e-mail servers have restrictions on the size of the incoming messages. V1 mitigated this problem by splitting large messages into several smaller ones, but this often caused a secondary problem of inbox clutter. This clutter was exacerbated when an e-mail user used more than one e-mail client to read mail. For example, an Outlook user might also use Outlook Web Access to read messages from a different computer. Since the DyCER software not be running on the other computer, all of the DyCER messages waiting in the inbox would show up in the inbox, achieving no useful purpose and confusing and annoying the user with lengthy downloads.

The result was a system that relied primarily on retries to ensure message delivery. This caused full inboxes for many users and a general discomfort with the use of DyCER.

Collaboration without related communications mechanisms makes the process of difficult. Most collaboration requires some conversation to occur before data sharing takes place. With the V1 approach, communications occurred outside the scope of the DyCER system; a user would have to send an e-mail or instant message to a set of recipients to let them know that about the intent to share application data with the group to let them know who was going to be in the group. It seemed natural to use DyCER itself for these messages, and the easiest solution was to add some instant messaging capabilities to DyCER.

Cross-platform implementations are key to broad usage of collaboration tools. This is perhaps the most interesting lesson learned from the V1 effort – collaboration systems must support everyone, and platform-independent applications remove a time-consuming barrier to collaboration: the need to reach consensus on tools. Minority system users in a group of collaborators determine what the majority of the group members uses. For example, a Mac user in a group of Windows users can veto the use of any software that is not available for a Mac. Sometimes the veto is unspoken and implicit, but the effect is just as strong. This is a very powerful position for the minority to be in and ultimately restricts group collaboration. This may be why web-based systems (blogs, wikis, etc.) are considered such promising possibilities for future collaboration systems. DyCER is even better, because it avoids the problem of administering a server.

These lessons became some of the driving factors behind DyCER Version 2, the most broadly distributed version of the software. We discuss V2 in the next section.

4.2 DyCER Version 2

4.2.1 Overview

DyCER Version 2 (V2) drew from the experiences of V1 and recast the group formation and file replication mechanisms into a cross-platform system that reached beyond SMTP to implement a broad set of collaboration functions. The underlying mechanisms in V2 draw from the store-and-forward approach of e-mail but implement that capability using a protocol that overcomes the limitations and problems of SMTP while offering a more efficient communications channel.

V2 implements the following core collaboration functions over the new protocol:

- Group file sharing
- Group discussion
- Group screen sharing
- One-to-one instant messaging
- Presence
- Direct file exchange (without group formation)

The sections that follow describe the architecture of V2 and the specific implementation details of the system.

4.2.2 Architecture

Architecturally similar to V1, DyCER V2 is implemented as a message passing system in a distributed topology similar to SMTP infrastructure. V2 broadens this architecture to allow for direct connection between user nodes where possible. This means that there are three forms of connectivity supported in the V2 architecture – direct, relayed, and SMTP (Figure 4).

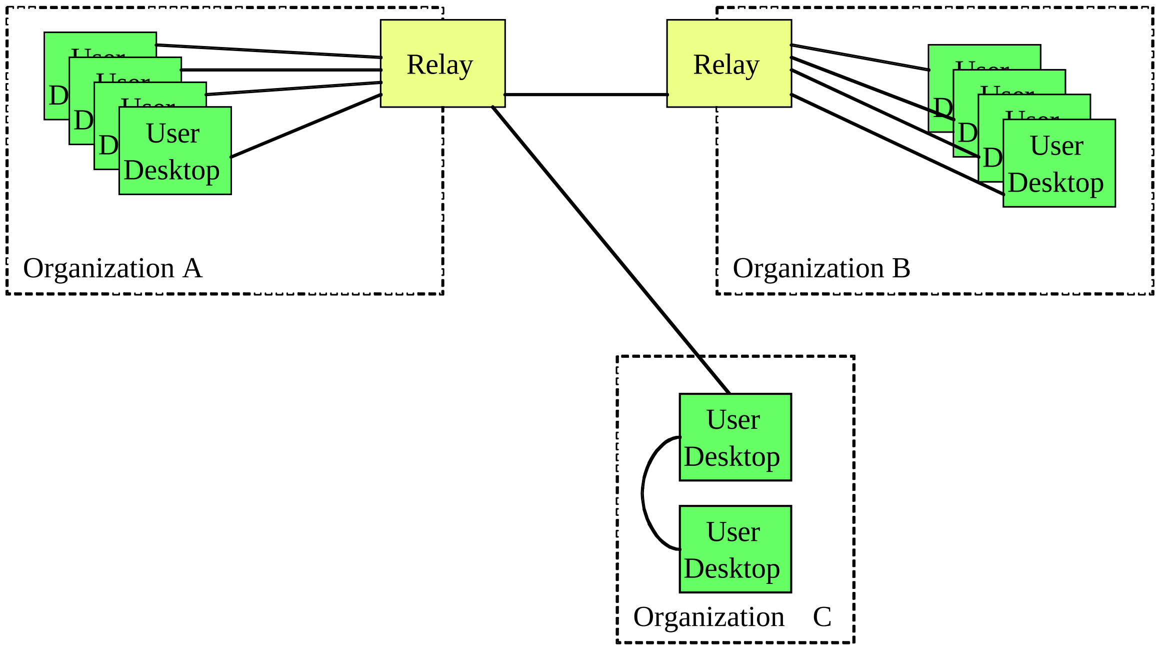


Figure 4 - Relay-based communication

The most basic connection mechanism is direct-connect (Figure 4, Organization C). It allows computers running in a single network "collision domain" to discover each other and to communicate. The mechanism is based on a broadcast message sent over a local subnet. This technique also works using wireless connections in ad hoc or peer-to-peer mode. Once connected, the system passes messages between the peers.

In cases where direct connections are not feasible, V2 implements a store-and-forward relay system (Figure 4, Organizations A and B). A user designates a relay as a "home" node. All messages from that user are sent to the relay node for routing to the intended recipient. Relay nodes may also connect to other relay nodes, allowing multiple organizations to share information without sharing a common relay. Relay nodes can store messages, thus accommodating users with intermittent connectivity.

In cases where relay nodes are impractical, V2 can also transmit messages via e-mail. E-mail can be sent from a client node using MAPI or SMTP. V2 clients can be configured to automatically retrieve messages from e-mail using Microsoft Outlook or any POP3 server. It should be noted, however, that this capability is not used, as all V2 users have opted for relayed and direct communications.

Nodes in V2 are not identified by e-mail address as in V1. Instead an identity is constructed by a combination of the relay name and a user-selected nickname. This has a similar look to an e-mail address, although a "!" character is used instead of an "@". For example, V2 identity might be "jeff!shinkuro.com". The first part, "jeff", maps to the cryptographic identifier that assures that only that user can read messages addressed to him (see section 4.2.4). The second part, "shinkuro.com, identifies the location of the user's "home" relay. This is where messages from other nodes should be routed if they cannot be delivered directly to the user. The user's nickname can be changed without consequence as long as it is unique with respect to the "home" relay; those changes are propagated to other users in the system.

Internally, the V2 software uses a layered architecture with single-instance managers responsible for core functionality (Figure 5). Message management and delivery is handled by the queue manager which delivers messages created by other managers in the system and receives and dispatches incoming messages from the communications manager. Above the queue manager, the internal functionality is broken into two main types of functionality – group-based functions (supported by the group manager) and individual (one-to-one) functions.

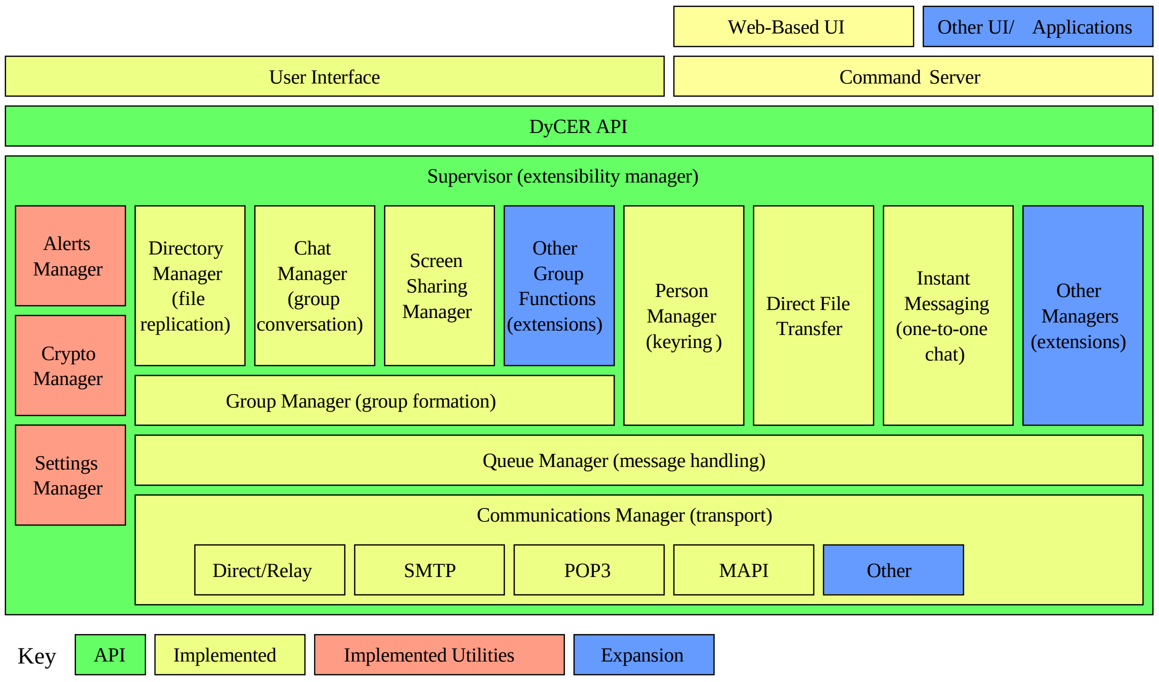

Web-Based UI
Other UI/ Applications
User Interface
Command Server
DyCER API
Supervisor (extensibility manager)
Alerts Manager
Crypto Manager
Settings Manager
Directory Manager (file replication)
Chat Manager (group conversation)
Screen Sharing Manager
Other Group Functions (extensions)
Person Manager (keyring)
Direct File Transfer
Instant Messaging (one-to-one chat)
Other Managers (extensions)
Group Manager (group formation)
Queue Manager (message handling)
Communications Manager (transport)
Direct/Relay
SMTP
POP3
MAPI
Other
Key
API
Implemented
Implemented Utilities
Expansion

Figure 5 - Software Modules

The user interface interacts through the supervisor with the managers in the system. The supervisor also has the responsibility of running the individual threads that drive the communications and message processing functions. This allows multiple applications to be constructed by linking to the DyCER API layer. In V2, two applications were constructed using this mechanism – one with a graphical user interface and one without. The GUI version is most typically run on client desktops while the application without the user interface typically runs as a relay.

To help expose the functionality of the DyCER core without requiring a linked application, V2 exposes a socket-based command server. With a protocol that looks similar to POP3 in style, the command server allows a single DyCER instance to be accessed by multiple user interfaces. For example, web browsers can access a web site that uses the socket to retrieve status information. A third party application could connect to issue a directed file transfer to a particular user. This allows any developer to connect a software application to DyCER without any special library requirements.

The sections that follow discuss several of the details of the V2 implementation as well as alternatives that can be explored in the future.

4.2.3 Data

DyCER communicates between nodes by sending messages. These messages are coded for a specific set of recipients and are passed from one relay to another until they arrive at their destinations. DyCER messages use SMTP's technique of having a clear identification of the recipients and the sender separated from the encoded message.

4.2.3.1 Format

4.2.3.1.1 Open Node Syntax (ONX)

The bulk of the data transmitted in the DyCER system is binary format, since virtually all of it is encrypted. This design decision rendered formats such as XML problematic, since binary data would need to be encoded in some way, such as base64. While base64 encoding does allow binary data to be packaged in XML, it also increases the size of the data by 25 percent; when replicating files that bulk can impact the transmission performance of the system.

To resolve this problem, while still using a tagged format, DyCER uses a format called Open Node Syntax (ONX). ONX is similar to XML in that it provides a hierarchical data structure where containers and values are tagged. However it improves on XML in the following ways:

1. End-tag names are optional. XML requires that beginning and ending tags be fully specified.
2. A tagged item may contain multiple values. For example, a list of values in XML would require that each value be individually tagged. ONX allows the values to be preserved as a list without additional tagging.

3. Values may contain binary data. Binary data may be used by escaping individual characters (there are only three of these escape codes) or by specifying a run length. The latter is particularly useful for lengthy binary data streams, where a few characters specifying the run length can encode a large amount of binary data.

A sample ONX message looks like this:

```
:message{
   :agent{
      :name["shinkuro"]
      :version["2.0.0.4513"]
      :date["2003-07-18 11:33:00"]
      :expires["never"]
   }
   :id["7bebed04-42c3-19ec-ad4d-306edacbab1b"]
   :dispatch["personmanager"]
   :verb["update"]
   :confirm["1"]
   :from["jeff!shinkuro-test.com" "520531e74bcd4fdb"]
   :arguments{
       :certificate["-----BEGIN CERTIFICATE-----
MIIDKTCCAhGgAwIBAgIBADANBgkqhkiG9w0BAQQFADAAMB4XDTAzMDcxNjAxNDEy
NFoXDTEzMDcxMzAxNDEyNFowXjEUMBIGA1UEAxMLSmVmZnJleSBLYXkxHzAdBgNV
BCkUFmplZmYhc2hpbmt1cm8tdGVzdC5jb20xJTAjBgkqhkiG9w0BCQEWFmplZmZA
mRWZOw9ImD1HUEVC/cUiSXQZJXBMx9Yu17GWYMRa3rTYtSK9p0gbkGQPlVHnbSfj
GM59iEKT/cQofbxSgQYBsNmFUcMzQx+HbE0OozxBt8o3OlVhY3RrKLcFno4ehH4t
VThgo6pCekrH3IiV/IFJo9V4viLRigCKZPyyYtltPWOrJKfyDO0U06DHDnce
-----END CERTIFICATE-----"]
      :picture["\[e2f].Ø....JFIF...........Û.C...........   .......
ÃÄÅÆÇÈÉÊ.ÓÔÕÖ×ØÙÚ....æçè....óô.......Ä............................
.Ä..................w.......!1..AQ.aq."2...B....      #3R..brÑ
ÉÉ...j...Èü.?y'...,..<K.j..ÇØç.Ìý..Êç.L.f^..û=CT...3Ì.ý]wz...Y..Ç..g.ûÏ
.Ç\F.yþ..È..ç.w...oy.ÉûÏ.I.AÑ.....ì.\y~d.y.ó...
.#.Ï.u..þ]Ä.....j.e..WN.uF*..^IæG.Ô.\Iq'.$.~g.*I?Ñì..Jì9L;.<Ë.*:?ÖQ]tÎs.
.{y%..,......}G.]Ç,QÉ.×É..ý..4.......ÕÇ..ÇæI
..,ë..üB....op14.....:.ý...Ð2Ó....l.y..IV?yQ..x.ü#.~g.Ë.....i....i.Ç..ý.
Ç$~g.#.RÛ]Ö!.....W../.z.ku.{.;....rÇ...|..W...=J....
lE:..Ìh...R..A.%...þ{~.Êx.P.9>Ï..g.^...Ø.5ko..$pn?y<.....ôÎþo?|l.ÅJ.....
.Û;O
.:.......Cû.W%......u...$.Ëó$......_Är}..Ã..1Ü...?..GÆ.....y..<...w...J.
:..o..~óýec.Üy..Ú#.ÛZ.).J.5..no.ýMgh@.ó2ì,...
....^y.}.:ÔÖ...Ç.Ç.Íë\.oO.È.ü.E.Wa...?..4.......d.Ë.....ryu......Èn$..$ó
Ôz../.Îg..È......þ!tÏK..8...yVü.çç.ü...Q
c.s..Ù"]
   }
}
```

The example shows a single message with multi- and binary- value elements. In ONX, curled braces (“{ }”) indicate container relationships while square brackets (“[ ]”) indicate values. Values are delimited with double-quotes. In the example above, the “from” tag contains two values. The “picture” tag contains a run-length escaped value – a JPEG image. The “\[e2f]” in the picture tag indicates the length of the escaped binary data run – 3631 bytes of binary data follow.

To facilitate the use of ONX, a streaming, event-based ONX parser was implemented, similar in fashion to James Clark’s EXPAT XML parser (expat.sourceforge.net). ONX can parse in “streaming” mode, accepting strings of arbitrary length. As the parser discovers containers and values, event handlers are called to process the data.

4.2.3.1.2 Future Directions

DyCER originally used XML for data encoding. After experiencing the difficulties inherent in XML, we switched to ONX in V2. While ONX has the distinction of being more efficient than XML, it is not a standardized format. Recently, the World Wide Web Consortium (W3C) released a recommendation called XML-binary Optimized Packaging (XOP -- http://www.w3.org/TR/2005/REC-xop10-20050125/), an acknowledgement to the problem of transmitting binary information in XML. XOP addresses the problem by using multipart/related MIME packages to contain the XML structure with references in the XML structure to the other parts of the MIME package. While this is functionally acceptable, it also forces the outer wrapper to be something other than XML. The outer wrapper would first have to be MIME processed with all of the pieces recognized and then the XML portion processed to access the binary information. We have not elected to use this representation.

4.2.3.2 Transport

Data transport has undergone two implementations. V1 used SMTP and e-mail retrieval protocols for transport, but V2 shifted to a new, relayable protocol. The following sections discuss these protocols.

4.2.3.2.1 E-Mail Protocols

The V1 architecture used SMTP and e-mail retrieval protocols for passing DyCER messages between network nodes. Messages were MIME-encoded. Depending on the user’s configuration, outbound messages were sent via SMTP directly to an e-mail server or via MAPI through Microsoft Outlook. Message recipients were identified by e-mail address and the same message could be routed to multiple recipients in the same fashion that regular e-mail messages are routed to multiple recipients. DyCER messages were marked with special tag in the subject header – [:/Shinkuro] – and an extra x-shinkuro header for SMTP messages. Because MAPI doesn’t directly address MIME and RFC-822 message encoding, messages sent via Outlook did not contain the extra header.

Receiving messages was a more complicated proposition. DyCER used one of three different mechanisms to accomplish this. The most basic approach was POP3. V1 could be set up to directly connect to a users POP3 server, where it would scan for DyCER messages. Messages with the designated header or subject line were retrieved and deleted from the inbox. This configuration was most common for individuals who had a separate e-mail account to use for DyCER.

The second configuration for message retrieval was a POP3 proxy. The V1 software could be configured to connect to a user's e-mail server when an e-mail client connected to the V1 software. The V1 software had a built-in POP3 server; a user would configure an e-mail client to connect to "localhost" with a user-id that included the location of the real e-mail server. For example, a user with an account on "shinkuro.com" with the ID "jdoe" would configure his client with the user-id "jdoe@shinkuro.com". The POP3 proxy in the V1 software would parse the user-id and then establish a connection to the real POP3 server. As messages were retrieved, the V1 software extracted and removed DyCER messages and processed them.

The third configuration for message retrieval was a Microsoft Outlook plug-in. As Outlook retrieved messages, a special process extracted the DyCER messages from the inbox and stored them in a pre-designated directory on the file system. V1 scanned the directory from time to time to pick up the messages and process them. As the messages were extracted to the file system, they were deleted from the inbox.

The problems with these approaches were discussed in section 4.1.3.

4.2.3.2.2 Relay Protocol

Because of these problems, V2 moved away from e-mail. Because the store and forward paradigm is so powerful for managing offline users, V2 implemented a mechanism that mirrored e-mail but with protocols that are stronger than e-mail.

V2 uses a subset of the BEEP (RFC3080) protocol to frame the messages that are transported from user node to relay node. Relay nodes accepting inbound messages store them on either on disk or in memory. Messages are assumed to require persistence unless marked as "online-only". Messages so marked only need to be delivered to nodes that are online; to improve performance for these messages, they are stored in memory instead of on disk. While the full BEEP protocol was not implemented, enough of it was to support the possibility of multiplexed streams of messages. These streams would allow higher priority messages to use greater bandwidth than lower priority messages.

Messages received by a node are acknowledged so that the sending node is completely aware of the completion of the transaction. Messages that cannot be fully transmitted can also be pushed back to the sender for retry or other purposes. In this way, full custodial transfer of messages is accomplished and the message is preserved in case of any difficulties.

Message addressing was accomplished through the key identifier/relay node combination identified earlier. A node transmitting a message first determines if the message can be delivered to a directly connected recipient. If not, the message is routed to a relay for delivery. A relay node might deliver a given message to another relay for final delivery to a recipient or hold the message until a recipient connects.

One of the unique capabilities of the V2 protocol is the Overcome By Events (OBE) mechanism. This allows a message, even if opaque, to be replaced before delivered. The OBE protocol uses an external key on a message to identify it. A message received with an OBE key replaces any existing message in the relay with the same OBE key. This mechanism is particularly useful for the screen sharing application – each screen shot is encoded in a message with an OBE key. Undelivered messages are replaced with newer ones; this creates an adaptive stream where higher bandwidth users receive more messages while lower bandwidth users receive fewer. But in all cases, each user receives the most up to date information possible.

4.2.3.2.3 Future Possibilities

One of the most obvious possibilities for DyCER is to develop a multiplexing mechanism where higher priority messages could move more quickly through the stream. This is particularly useful for applications like screen sharing.

Another opportunity is to add a prioritization mechanism to the protocol. This would allow more important messages to be sent before less important ones. This might be useful for updates that have to travel over lower bandwidth channels.

4.2.3.3 Storage

Because messages must be allowed to persist, each DyCER node contains an embedded database. This embedded database stores the metadata for each message, but the message itself is stored as a file on the disk. This allows for relatively high performance access when the messages are scanned for delivery opportunities.

4.2.4 Security

DyCER implements all its communications functions using a self-contained security system that can be connected to other external security systems. Each installation of the software generates a 2048-bit RSA key pair and a corresponding self-signed certificate (X.509v3). The least significant 64 bits of the public key are used as a node identifier, the value that uniquely identifies each software installation. When nodes communicate, the self-signed certificates are exchanged to establish the identity and basis of security for further information exchanges.

DyCER has “end-to-end” data protection. Since communications may pass through more than one node, messages are individually encrypted and signed instead of using a transport level security such as secure sockets layer (SSL). A unique key is generated for each message transmitted and each message is encrypted using 256-bit AES. In addition, each message is signed by the sender the ensure authenticity and that no tampering has occurred.

4.2 5 Groups

The core purpose of the DyCER software is to implement small group collaboration through using peer group mechanism. The following paragraphs describe the application-level protocol that implements this functionality.

4.2.5.1 Group Formation Protocol

DyCER uses a decentralized peer strategy for group formation. A globally unique identifier (GUID) is generated as the identity of a group (Figure 6). Membership to a group is designated by possession of a certificate containing that identifier and signed by another group member. To be recognized as a full member of a group, an individual must possess a certificate signed by every other group member who possesses the group GUID. This unconventional technique offers the unique ability of establishing peer groups without the need for any centralized mediation.

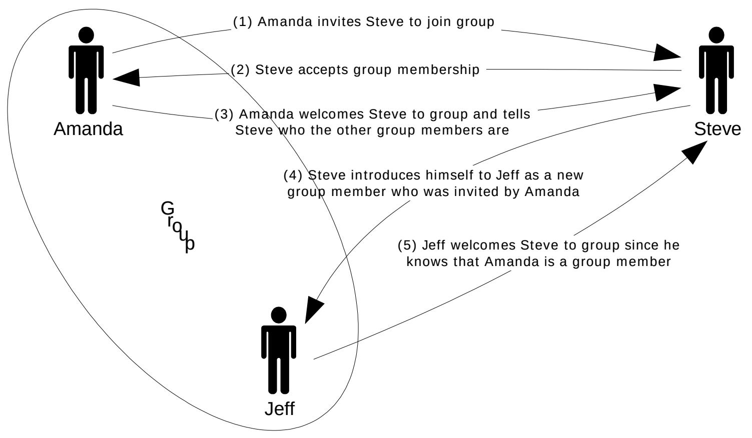

(1) Amanda invites Steve to join group
(2) Steve accepts group membership
(3) Amanda welcomes Steve to group and tells Steve who the other group members are
Amanda
Steve
(4) Steve introduces himself to Jeff as a new group member who was invited by Amanda
Group
(5) Jeff welcomes Steve to group since he knows that Amanda is a group member
Jeff

Figure 6 - Group invitation protocol

Membership by exchange of certificates between group members is best described with an example (Figure 6). When Amanda invites Steve to join the group, Amanda sends her certificate to Steve. Steve signs a certificate containing Amanda's public key and the group identifier and sends it back. Amanda then signs a certificate containing Steve's public key and the group identifier and sends it to Steve along with a list of other group members. Steve then "introduces" himself to the other group members by presenting the certificate signed by Amanda as well as a certificate he signs containing the public key of each other group member and the group designation. Each other group member responds by issuing a corresponding certificate.

The association of rights and privileges further enhances this basic group formation mechanism. Because the formation is entirely through cross certification of members, the invitation process also includes an exchange of requested privileges. When Amanda invites Steve to join the group, Amanda's invitation includes the set of group privileges that she has within the group. When Steve signs the certificate with Amanda's public key, it also includes this list of privileges. Similarly, Amanda transmits the privileges that Steve is being offered as part of the invitation. When Steve accepts membership and returns the signed certificate, Amanda then returns a signed certificate back to Steve with his assigned privileges. When evaluating whether to welcome the new member to the group, others within the group can evaluate the privileges identified as well as the privileges of the signer of the new member's certificate to determine whether the new member should be in the group.

The effect of this strategy is the creation of a pure peer formed group using cross-certification to manage the process. Revocation of group membership occurs in a similar fashion. Variations on this could include the use of PKI to further authenticate the group members (currently, self-signed certificates are used to identify individuals).

One of the key benefits to a strategy like is that it closely models how organizations are formed in the "real world". While not all "real world" groups require full introductions as defined in the above example, neither does DyCER.

The newest versions of DyCER implement a mechanism that allows group members to be either "full" or "limited" members. Full members are treated as pure peers, where all full members are introduced to each other. Limited members are only introduced to full members, but not other limited members. This asymmetric mechanism provides the means for restricting the distribution of information. Furthermore, access rights are assigned in the certificates, making it possible to restrict members to read-only access.

4.2.5.2 File Replication

DyCER uses the group formation mechanism to implement group file sharing through replication. When a group is created, a file folder is associated with it. The contents of that folder are replicated into the folders of the other group members.

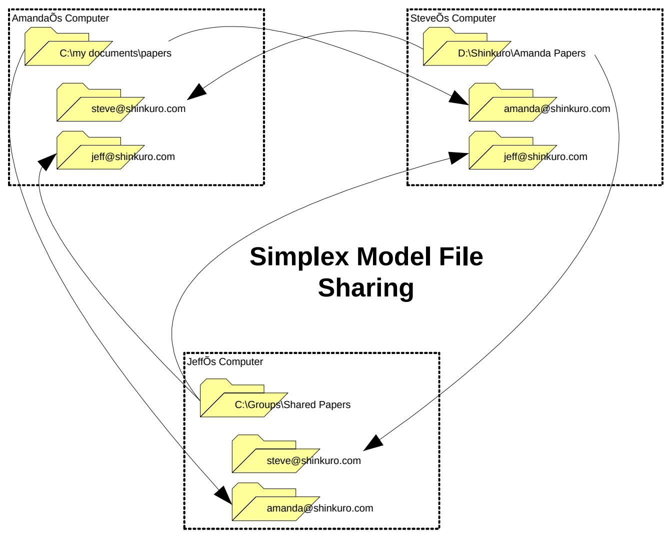


Figure 7 - Simplex Model File Sharing

To avoid conflict resolution, DyCER implements a "simplex replication system" that avoids the problems of simultaneous updates (Figure 7). Replicated files from other group members are placed into separate subdirectories of the shared folder. This allows multiple version of the same file to exist. These replicated files are marked read-only, preventing changes on the local machine. To change a file, it can be "promoted", which removes the read-only flag and copies the file into main group folder.

To ensure that a user does not need to search through a set of directories for a file, DyCER software renders a view of the group files, showing which files are identical and which ones are different. This view shows who contributed the file to the group as well as whether the file has replicated to all other group members.

4.2.5.3 Group Discussion

One of the other services built on the group formation functionality is group discussion. This facility offers multi-user chat capability, with the contents of the discussion distributed to all group members, whether or not they are on-line. The discussion remains persistent for the group, allowing it to serve as a conversation facility as well as an activity log.

4.2.5.4 Screen Sharing

In addition to group discussions, DyCER software also implements a group screen sharing facility that allows a group member to broadcast the contents of his or her computer screen to other group members. This capability captures the bits on the screen and broadcasts them to all group members. When a group member chooses to transmit, a list of available broadcasts is listed for the other group members. A group member then can select which display to receive, which generates a message back to the broadcaster to add that recipient to the list of group members that should receive the broadcast.

The screen bits are captured and transmitted as a stream of message to the list of "tuned-in" recipients. This, in effect, uses a multicast technique to transmit the screen image. However, since the connection speed of recipients may vary, the broadcast protocol takes advantage of a DyCER's unique OBE technique.

OBE is a mechanism that allows outbound messages to be overtaken by newer messages, obsoleting and deleting the old ones. This technique allows the screen to be transmitted as frequently as possible by the broadcaster while the recipients are assured of receiving as recent an update as is possible based on their connection speed. OBE works by adding a series of tags to the outer wrapper of a message that describes the content without revealing it. Any subsequent message declares an older message with the same tag as obsolete and replaces it.

4.2.6 Individuals

With the creation of V2, the higher performance protocol offered the opportunity for presence, instant messaging, and other one to one messaging functionality. This section describes these capabilities.

4.2.6.1 Presence

Presence is handled in two different ways because of the two different communications capabilities. For direct connections, presence is handled by tracking the actual socket-based connection between nodes. However, nodes that communicate via relays have to rely on the relay to establish presence for them. Each relay is responsible for maintaining a presence map for each of its users. The map shows the user and the nodes that it wishes to monitor. For presence to be established between two users, they must list each other on their lists of monitored users. Only then will a relay identify a user as being online.

For nodes that are separated by two relays, the process is similar, although neither relay trusts the other completely to establish presence. Therefore, a relay will establish a "proxy" connection for a given user to the other relay. This allows the user to send the presence request directly (and privately) to the other relay, which then privately sends the response. The proxy connection scheme avoids many trust problems between nodes.

4.2.6.2 Instant Messaging and Direct File Transfer

Instant messaging and direct file transfer are built on the same mechanism. Each generates one or more messages to send directly to a recipient. One of the unusual elements in DyCER, however, is that instant messaging does not necessarily require the recipient to be online. Messages are queued and held until the recipient connects; at that time the messages are delivered.

Direct file transfer was developed to enable users to send files to each other quickly without requiring the formation of a group. It essentially packages a file like an instant message and transmits it to a given recipient.

4.2.6.3 Secure E-Mail

With the V1 experience in e-mail, building a secure e-mail system within DyCER was a natural next step. POP3 and SMTP servers were added in such a way that e-mail clients could connect to the DyCER client software as if it were an e-mail server. The e-mail client connects to local host for both POP3 and SMTP services. When an e-mail message is sent from the e-mail client, the DyCER SMTP server receives it and packages it for a DyCER recipient, including the full complement of encryption and signing. From there, the message moves through the regular DyCER communications paths until it reaches the recipient's node. On that node, the DyCER software makes the message available to the POP3 server. When the e-mail client polls for POP3, the message is extracted and delivered.

One of the key advantages of this technique is the security. Merely by having a DyCER node, one can get secure e-mail messaging to other DyCER users. In addition, the DyCER protocol is more robust than e-mail protocols, offering opportunities for full acknowledgement, message tracing and delivery.

4.2.6.4 Social Networking

To encourage usage of the DyCER software, V2 implemented a social networking capability. This allows a user to ask another user for a list of their friends. If enabled on the queried side, the user responds with an instant message containing that list of friends. The originator can then add these individuals to their list of friends and request that the others do the same. This allows first-time users to find others quickly, especially by asking a DyCER relay for its list of friends.

4.2.7 Application Programming Interface

Because DyCER is an active application with a single node identity, it is impractical for multiple copies of the software to be running for a single user. Instead, DyCER works better by running a single instance with a mechanism for other applications to connect. The DyCER API accomplishes this with a POP3-like protocol through a command server. The command server interface was initially used to implement a web-based interface for the daemon version of the software. Applications can use this interface to create groups, send files and instant messages, and acquire presence information.

4.2.8 Cross-Platform Implementation

DyCER is designed as a cross-platform open system. The core functionality of the software is built as a stand-alone library, which can be incorporated into other applications. Further, the library implements a module system in which new functionality can be built. This library, as well as the current client software runs on Microsoft Windows, Mac OS X, FreeBSD, and Linux.

4.2.9 Distribution and Usage

4.2.9.1 Statistics

Both V1 and V2 were certified for mass-market distribution and as such were released to the general public via the Shinkuro website (www.shinkuro.com). V1 received limited usage, as many of the issues associated with e-mail were a problem for most users to handle. Those issues and many others were resolved with the release of V2.

V2 was initially released on the Shinkuro website in March 2004. To better track distribution, the Shinkuro website included a mechanism request registration details (including a valid e-mail address) and an e-mail reply with a registration code to activate the screen sharing and subdirectory replication features. As of February 2005, there were approximately 1,800 downloads of the V2 software with just over 1,300 unique e-mail addresses.

From there, the usage of the software could be monitored based on those who connected to the open shinkuro.com relay, provided to those who were not interested in setting up their own relay. Usage of the software fluctuated over time, but the maximum usage of the relay peaked around 80 concurrent users. Log files for the relay generally indicated about twice that number of users connected sometime during a 24-hour period.

Over the year, several users set up relay nodes. Shinkuro set up two relays as part of a beta program. One of those systems is currently in place and has a potential of over 50 users. Currently that system sees about 12 concurrent users on a regular basis. The other relays appear to be inactive, likely because the daemon version of the DyCER software has not been officially released. Currently that version runs the main shinkuro.com relay and one other relay in beta mode.

The general feedback form the current user community has been overwhelmingly positive, with several suggestions for improvements of the user interface and functionality. Where possible, these suggestions have been investigated and implemented; a large set of these ideas still exists.

Several dedicated users have been with the system since the early days and continue to use it regularly. However, user adoption has been slow in spite of efforts by the Shinkuro team to encourage adoption among colleagues, friends, and family. It is possible that, with a broader marketing campaign or a specific vertical market, the DyCER software could make inroads into the market, but as of this point it has not.

4.2.9.2 The Network Effect

One of the most sought-after software distribution dynamics is the network effect. The network effect in the case of DyCER is the viral spread of the software, creating a huge user community. This effect has been seen in several other software packages – e-mail and instant messaging. A user community builds upon itself by virally spreading when users encourage other users to adopt and use the software.

Originally, we expected this to lead to mass adoption of the software. This is based on the notion that as users are invited to a group, they, in turn, create new groups and invite new users. The resulting network should grow exponentially for a period of time until it the bulk of users end up connecting to existing users.

However, the network effect has not occurred yet with DyCER. While it still might, it is worth exploring why it has not happened yet. Conventional wisdom would dictate that the software either has usage difficulties or lacks compelling value. Neither of these factors has prevented a strong following by a small group of users. In fact, the industry's repeated attempts to create a compelling set of collaboration tools indicate that the market does see the value of the software.

Instead, there may be a subtler factor dampening the network effect. The reality is that group dynamics are different than communications dynamics. The spread of communications facilities (e-mail and instant messaging for example) is driven by a powerful need for humans to talk with each other. These facilities have become electronic versions of the hand-written letter or memo in many cases. However, groups are generally formed by a much smaller set of leaders, with the followers becoming group members. For example, an individual might form a group with ten members, but only one of those members might be in the position to need to form groups with other members. Arguably, there is a significantly lower percentage of individuals who form groups (call meetings, etc) than there are who merely attend them. This means that out of a given number of DyCER users, a very small number of them might actually reach out to others to use the software. Further, groups formed within DyCER have a limited lifetime (generally), which may result in a drop-off in usage. While some attempts to mitigate this have been implemented, most notably the chat component of DyCER, instant messaging does not seem to be sufficient to compete with other IM tools on the market and thus drive the usage of DyCER.

4.2.10 Future Activities

4.2.10.1 First Responder Appliance

The First Responder Appliance is an appliance packaged with the DyCER software, designed for use by emergency situation responders.

4.2.10.1.1 Background

Homeland security and public safety operations require secure group collaboration and information exchange technologies to support mobile command post and incident site activities. The most likely scenarios involve the ad hoc assembly and integration of national, regional and local responders and management agencies into effective teams that must address an unpredictable range of incident consequences. Dynamic team composition, portability and mobility of collaboration systems will be required.

Facilities with electrical power (either from utility companies or contingency generators) may or may not be immediately available to homeland security responders. This system addresses short-term lack of available electrical power by designing the system to run on a self-contained battery supply until requisite contingency infrastructure is established.

Members of these ad hoc teams will likely arrive with a variety of hardware and software platforms – Linux, Mac, Windows, Windows for Handhelds, and other units. As team members arrive at a given site, they connect to the wireless access point/appliance, connect via the web to the First Responder appliance and download the collaboration tools. Once the software is installed, secure group information sharing activity is enabled, including secure email, IM, file sharing, and application/desktop sharing.

4.2.10.1.2 Description

The First Responder Appliance is a wireless networking (802.11 b/g) router that includes DHCP, HTTP and email services that can support the rapid implementation of a common suite of secure group collaboration tools for dynamic and ad hoc team formation in fixed facilities or sparse remote site operations. The appliance is based on Linux. Initial units will require power, but battery-powered units are envisioned. It is designed to be operated as a stand-alone unit, but can be integrated into a larger network environment

The First Responder Appliance will connect to a broad set of devices will cover a more encompassing set of field operations, including collection of digital photos, fill-in forms and the like, and may include one or more of the following units:

- Pocket PC Devices
- Tablet PCs
- Linux-based notebooks
- Windows-based notebooks

4.2.10.2 JWID Participation (June 2004)

June 2004 was devoted almost entirely to participation in the Joint Warrior Interoperability Demonstration (JWID) 2004. Shinkuro played a significant role in this activity with installations of two relays and approximately 40 clients across 4 sites – NORTHCOM (Colorado Springs), SPAWAR (San Diego), DISA Eagle (Arlington, VA), and Dahlgren. Shinkuro was used in the demonstrations as a means to share twice daily status briefings, network status displays, training, and technical support. Several of the other DARPA contractors that participated in the exercise used Shinkuro on site and at their own offices to update software and configuration files for their own demos also.

4.2.10.3 Alternative implementations

One of the possibilities that has emerged from the DyCER effort is that it is possible to implement the functionality into other environments. The most likely possibility is that of the Jabber environment, which has been examined as XML Tactical Chat (XTC) within the DoD. Although Jabber's protocol is currently incompatible with DyCER, it employs many of the same architectural concepts. The basic group formation protocols implemented in DyCER could be modified to run on XMPP (the Jabber protocol), although some significant effort would have to go into the file replication protocols. The key problem with XMPP is that it is basically XML, which, as described earlier, incurs a 25 percent penalty for transmitting binary data. Jabber servers suffer from another limitation because they do not have the store and forward capability that DyCER relays do.

Nonetheless, because of the similarity of the architectures, if XTC becomes the standard throughout the military, the DyCER file replication and security functionality could be integrated with it.

5.0 Conclusions

DyCER's powerful group file sharing capability successfully addressed the basic concepts of the Dynamic Coalitions program by creating a mechanism for allowing coalition partners to securely share information without requiring centralized infrastructure. The decentralized, distributed architecture allows the software to scale significantly as the user community grows.

While not yet a commercial success, DyCER has the potential to become a stronger player in the collaboration space by continuing to promote its cross-platform approach. DyCER needs to be ported to additional platforms (such as NetBSD and handheld units) to completely address this. In addition, formal review of the cryptographic components of the software to certify the security of the system would be helpful in approaching a government marketplace.

List of Symbols, Abbreviations, and Acronyms

Term	Expansion
AES	Advanced Encryption Standard
API	Application Programming Interface
BEEP	Blocks Extensible Exchange Protocol Core
DHCP	Dynamic Host Configuration Protocol
DyCER	Dynamic Collaboration Environment Research
FTP	File Transfer Protocol
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
MAPI	Messaging Application Program Interface; also called Mail Application Program Interface
MIME	Multimedia Internet Mail Extensions; also called Internet Mail Exchange, Multimedia Extensions, Multipurpose Internet Mail Extensions, Internet Mail Extensions, and Multimedia Internet Mail Exchange
NAT	Network Address Translation
OBE	Overcome by Events
ONX	Open Node Syntax
P-2-P	Peer-to-peer
POP3	Post Office Protocol
RSA	An Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman
SMTP	Simple Mail Transfer Protocol; also called Simple Mail Transport Protocol
SSL	Secure Socket Layer
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language